Postel

Network Working Group

Request for Comments #57

Mike Kraley (Harvard)

John Newkirk (Harvard)

June 19, 1970

Thoughts and Reflections on NWG/RFC #54

In the course of writing NWG/RFC #54 several new ideas became apparent. Since these ideas had not previously been discussed by the NWG, or were sufficiently imprecise, it was decided not to include them in the official protocol proffering. We thought, however, that they might be proper subjects for discussion and later inclusion in the second level protocol.

I. Errors and Overflow

In line with the discussion in NWG/RFC #48, we felt that two types of errors should be distinguished. One is a real error, such as an RFC composed of two send sockets. This type of error can only be generated by a broken NCP. In the absence of hardware and software bugs, these events should never occur; the correct response upon detection of such an event was outlined in the description of the ERR command in NWG/RFC #54.

The other "error" is an overflow condition arising because finite system resources are exhausted. An overflow condition could occur if an RFC was received, but there was no room to create the requisite tables and queues. This is not a real error, in the sense that no one has done anything incorrect (except perhaps the system planners in not providing sufficient table space, etc.) Further, a

recovery procedure can be well defined, and simply entails repeating the request at a future time. Thus, we believe an overflow condition should be distinguished from a real error.

In NWG/RFC #54 an overflow condition was reported by returning a CLS, as if the connection had been refused. This sequence performs the necessary functions, and leaves the connection in the correct state, but the initiating user is misinformed. He is deluded into thinking that he was refused by the foreign process, when, in fact, this was not the case. In certain algorithms this difference is crucial.

In further defining error conditions, we felt that it would be helpful to specify why the error was detected, in addition to specifying what caused the error. While writing the psuedo-Algol program mentioned in NWG/RFC #55 we differentiated 9 types of errors (listed below). We would, therefore, like to propose the extension of the ERR message to include an 8-bit field following the op code to designate the type of error. This would be followed by the length and text fields, as before. We propose these error types;

0. UNSPECIFIED ERROR
1. HOMOSEX (invalid send/rcv pair in an RFC)
2. ILLEGAL OP CODE
3. ILLEGAL LEADER (bad message type, etc.)
4. ILLEGAL COMMAND SEQUENCE
5. ILLEGAL SOCKET SPECIFICATION - COMMAND
6. ILLEGAL COMMAND LENGTH (last command in message was too short)
7. CONNECTION NOT OPEN - DATA
8. DATA OVERFLOW (message longer than advertised available buffer space)
9. ILLEGAL SOCKET SPECIFICATION - DATA (socket does not exist)

In light of the other considerations mentioned earlier, we would also like to propose an additional control command to signify overflow:

OVF	my socket	your socket

The format of the message is similar to that of the CLS message, which it replaces in this context. The socket numbers are 32 bits long and correspond to the socket numbers in the RFC which is being rejected. The semantics of an incoming OVF should be identical to an incoming CLS; in addition, the user should be informed that he has not been refused but rather has overtaxed the foreign host's resources.

An alternative to creating a separate control command can be realized by considering the similarity between a CLS and an OVF. Conceivably, an eight-bit field could be added to the CLS command to define its derivation. We believe, however, that this alternative is conceptually inferior and practically more difficult to implement.

Overflow does not require serious consideration if it is a significantly rare occurrence. We do not believe this will be the case, and we further believe that its absence will be an unnecessary restriction upon the user.

II. Host Up and Host Down

Significant problems can arise when a host goes down and then attempts to restart. Two cases can easily be distinguished. The first is a "soft" crash, where the system has prior notice that the machine is going down; sufficient time is available to execute pre-recovery procedures. The other case can be termed a "hard" crash, often the result of a system failure. Insignificant warning is usually given; but more important, the state of the machine after recovery is rarely predictable.

When a host returns from a hard crash, the network will be in an undefined state. Very probably the NCP's data structures are destroyed or are meaningless. The network has declared the host dead--but only to processes which attempted data transmission and were refused. The only alternative for the crashed host is re-initialization of its tables. What are the alternatives for the foreign hosts?

We would like to propose the addition of two control commands: RESET (RST) and RESET REPLY (RSR). Each would consist solely of an op code with no parameters. Upon receipt of an RST, a host would immediately terminate all connections with the sending host, but would not issue any CLS's. The receiver of the RST would also note that the originator of the RST was alive, and would then echo an RSR to the sender. When a host receives an RSR, he should then note that the echoing host is alive. (The function of RST can be partially simulated if a host will immediately close all relevant table entries upon discovering that another host is down.)

Thus, after a hard crash, all connections and request for connections are terminated. The RST also informs all foreign hosts that we are again alive, and an RSR is received from every functioning

NCP. A host live table (see NWG/RFC #55) can easily be assembled, and establishment of connections can resume.

Related problems also crop up when we consider attempting to synchronize the network, which may still be carrying messages generated prior to the crash, with an NCP which has an initialized environment. We lack the facilities for unblocking links, discarding messages, etc.--facilities which this proposal will necessitate. Further interaction with BBN should resolve these difficulties.

The problems associated with "soft" crashes are not nearly as pressing, and they demand more sophisticated (i.e., complex) solutions. Our preliminary experimentation with the network demonstrates that a good initialization and recovery protocol are far more necessary.

Many of the ideas presented herin were germinated and/or jelled through conversations with Steve Crocker and Jon Postel. We would also like to acknowledge the assistance of Jim Balter and Charles Kline of UCLA, who devoted a great deal of effort toward helping develop the pseudo-Algol program which was the predecessor of much of our recent documentation.